Gestion d'évènements

Récupération du clique souris

```
import numpy as np
import matplotlib.pyplot as plt

class gestionnaire_evenements:

    def __init__(self,figure):
        self.figure=figure

    def connect(self):
        self.press=self.figure.canvas.mpl_connect("button
        _press_event",self.click_souris)

    def click_souris(self,evenement):
        print("click souris")
        print(evenement)


fig=plt.figure()
gestionnaire=gestionnaire_evenements(fig)

gestionnaire.connect()

plt.show()
```

Récupération du clique souris

```
import numpy as np
import matplotlib.pyplot as plt

class gestionnaire_evenements:

    def __init__(self,figure):
        self.figure=figure

    def connect(self):
        self.press =
        self.figure.canvas.mpl_connect("button_press_event",self.clique_
        souris)

    def clique_souris(self,evenement):
        x,y=evenement.xdata,evenement.ydata
        print("clique en ",x,y)


points=np.array([[0,1],[2,3],[4,7],[9,5],[5,5],[-2,3],[7,-1.1]])
fig=plt.figure()

gestionnaire=gestionnaire_evenements(fig)

gestionnaire.connect()

plt.plot(points[:,0],points[:,1],'o')
plt.axis([-5,10,-5,10])
plt.show()
```

coordonnées du clique

on affiche des points

Récupération du clique souris

```
import numpy as np
import matplotlib.pyplot as plt

class gestionnaire_evenements:

    def __init__(self,figure,points):
        self.figure=figure
        self.points=points

    def connect(self):
        self.press = self.figure.canvas.mpl_connect("button_press_event",self.clique_souris)


    def clique_souris(self,evenement):
        x,y=evenement.xdata,evenement.ydata
        distance_vectoriel=self.points-np.array([x,y])
        distance=(distance_vectoriel[:,0]**2+distance_vectoriel[:,1]**2)**0.5
        index_min=np.argmin(distance)
        if distance[index_min]<0.5:
            print("clique point ",index_min)
        else:
            print("clique sur aucun point")


points=np.array([[0,1],[2,3],[4,7],[9,5],[5,5],[-2,3],[7,-1.1]])
fig=plt.figure()

gestionnaire=gestionnaire_evenements(fig,points)

gestionnaire.connect()

plt.plot(points[:,0],points[:,1],'o')
plt.axis([-5,10,-5,10])
plt.show()
```

la classe à accès aux points

recherche du point le plus proche

Affichage interactif

```
class gestionnaire_evenements:

    def __init__(self,figure,points):
        self.figure=figure
        self.points=points
        self.point_selectionne=-1

        self.affichage()

    def connect(self):
        self.press = self.figure.canvas.mpl_connect("button_press_event",self.clique_souris)

    def clique_souris(self,evenement):
        x,y=evenement.xdata,evenement.ydata
        distance_vectoriel=self.points-np.array([x,y])
        distance=(distance_vectoriel[:,0]**2+distance_vectoriel[:,1]**2)**0.5
        index_min=np.argmin(distance)
        if distance[index_min]<0.5:
            print("clique point ",index_min)
            self.point_selectionne=index_min
            self.affichage()
        else:
            print("clique sur aucun point")
            self.point_selectionne=-1
            self.affichage()

    def affichage(self):
        plt.clf()
        plt.plot(points[:,0],points[:,1],'bo')

        idx=self.point_selectionne
        if idx>=0:
            plt.plot(points[idx,0],points[idx,1],'ro')

        plt.axis([-5,10,-5,10])
        plt.draw()
```

```
points=np.array([[0,1],[2,3],[4,7],[9,5],[5,5],[-2,3],[7,-1.1]])
fig=plt.figure()

gestionnaire=gestionnaire_evenements(fig,points)

gestionnaire.connect()

plt.show()
```

nouvel affichage

affichage à partir de la classe

Drag and drop

```
class gestionnaire_evenements:

    def __init__(self,figure,points):
        self.figure=figure
        self.points=points
        self.point_selectionne=-1

        self.affichage()

    def connect(self):
        self.press =
        self.figure.canvas.mpl_connect("button_press_event",self.cliqu
        e_souris)
        self.motion =
        self.figure.canvas.mpl_connect("motion_notify_event",self.depl
        acement_souris)

    def clique_souris(self,evenement):
        x,y=evenement.xdata,evenement.ydata
        distance_vectoriel=self.points-np.array([x,y])
        distance=(distance_vectoriel[:,0]**2+distance_vectoriel[:,1]**
        2)**0.5
        index_min=np.argmin(distance)
        if distance[index_min]<0.5:
            print("clique point ",index_min)
            self.point_selectionne=index_min
            self.affichage()
        else:
            print("clique sur aucun point")
            self.point_selectionne=-1
            self.affichage()
```

```
def affichage(self):
    plt.clf()
    plt.plot(points[:,0],points[:,1],'bo')

    idx=self.point_selectionne
    if idx>=0:
        plt.plot(points[idx,0],points[idx,1],'ro')

    plt.axis([-5,10,-5,10])
    plt.draw()

def deplacement_souris(self,evenement):
    if evenement.button ==1 and self.point_selectionne!=-1:
        x,y=evenement.xdata,evenement.ydata
        points[self.point_selectionne,0]=x
        points[self.point_selectionne,1]=y
        self.affichage()
```